08

Mãos na massa

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

1) Na classe `ContaCorrente`, altere todas propriedades para serem **privadas** (`Private`):

```
Private Extrato As String
Private Conta As Integer
Private Agencia As Integer
Private Titular As Cliente
Private Saldo As Double = 100
```

2) Altere também os nomes de todas as propriedades, adicionando um `m_` antes dos seus nomes:

```
Private m_Extrato As String
Private m_Conta As Integer
Private m_Agencia As Integer
Private m_Titular As Cliente
Private m_Saldo As Double = 100
```

3) Ao fazer isso, diversos erros ocorrem na aplicação, pois, além de terem seus nomes mudados, as propriedades também ficaram privadas. Para isso, existem o **`Get`**, para acessar uma propriedade privada, e o **`Set`**, para modificar uma propriedade privada. No Visual Basic .NET, no **`Property`**, há uma maneira muito simples de representar o **`Get`** e o **`Set`**:

```
Private m_Extrato As String
Public Property Extrato As String
    Get
        Return m_Extrato
    End Get
    Set(value As String)
        m_Extrato = value
    End Set
End Property
```

4) Agora, crie os respectivos `Property` para o restante das propriedades da classe, e declare os seus **`Get`** e **`Set`**. Além disso, separe o código com a diretiva `Region`:

```
Imports _2___ByteBank.ByteBank

Public Class ContaCorrente

#Region "PROPRIEDADES"

    Private m_Extrato As String
    Public Property Extrato As String
        Get
            Return m_Extrato
        End Get
```

```
            Set(value As String)
                m_Extrato = value
            End Set
        End Property

        Public Property Conta As Integer
        Public Property Agencia As Integer

        Public m_Titular As Cliente
        Public Property Titular As Cliente
            Get
                Return m_Titular
            End Get
            Set(value As Cliente)
                m_Titular = value
            End Set
        End Property

        Private m_Saldo As Double = 100
        Public Property Saldo As Double
            Get
                Return m_Saldo
            End Get
            Set(value As Double)
                If value < 0 Then
                    m_Saldo = 0
                Else
                    m_Saldo = value
                End If
            End Set
        End Property

    #End Region

    #Region "MÉTODOS"

        Public Function Sacar(ValorSacar As Double) As Boolean

            Dim Retorno As Boolean
            If m_Saldo < ValorSacar Then
                Retorno = False
            Else
                m_Saldo -= ValorSacar
                Retorno = True
            End If
            Return Retorno

        End Function

        Public Function Transferir(ValorTransferencia As Double, ByRef ContaDestino As ContaCorrente) As

            Dim Retorno As Boolean
            If m_Saldo < ValorTransferencia Then
                Retorno = False
            Else
                m_Saldo -= ValorTransferencia
                ContaDestino.Depositar(ValorTransferencia)
                Retorno = True
```

```
            End If
            Return Retorno

        End Function

        Public Sub Depositar(ValorDepositar As Double)
            m_Saldo += ValorDepositar
        End Sub

    #End Region

    #Region "FUNÇÕES ESPECIAIS"

    #End Region

    End Class
```

5) Agora, na classe `Cliente`, adicione o método `TestaCPF`, que você baixou no início da aula:

```
    Private Function TestaCPF(CPF As String) As String
        Dim dadosArray() As String = {"11111111111", "22222222222", "33333333333", "44444444444",
            "55555555555", "66666666666", "77777777777", "88888888888", "99999999999"}
        Dim vResultado As Boolean = True
        Dim i, x, n1, n2 As Integer
        CPF = CPF.Trim
        For i = 0 To dadosArray.Length - 1
            If CPF.Length <> 11 Or dadosArray(i).Equals(CPF) Then
                vResultado = False
            End If
        Next
        If vResultado = False Then
            Return "00000000000"
        Else
            For x = 0 To 1
                n1 = 0
                For i = 0 To 8 + x
                    n1 = n1 + Val(CPF.Substring(i, 1)) * (10 + x - i)
                Next
                n2 = 11 - (n1 - (Int(n1 / 11) * 11))
                If n2 = 10 Or n2 = 11 Then n2 = 0

                If n2 <> Val(CPF.Substring(9 + x, 1)) Then
                    vResultado = False
                    Exit For
                End If
            Next
            If vResultado = False Then
                Return "00000000000"
            End If
        End If
        Return CPF
    End Function
```

6) Ainda na classe `Cliente`, faça o mesmo que foi feito na classe `ContaCorrente`. Além disso, valide o CPF no seu `Set`:

```
Namespace ByteBank

    Public Class Cliente

#Region "PROPRIEDADES"

        Private m_Nome As String
        Public Property Nome As String
            Get
                Return m_Nome
            End Get
            Set(value As String)
                m_Nome = value
            End Set
        End Property

        Private m_CPF As String
        Public Property CPF As String
            Get
                Return m_CPF
            End Get
            Set(value As String)
                m_CPF = TestaCPF(value)
            End Set
        End Property

        Public Property Profissao As String
        Public Property Cidade As String

#End Region

#Region "MÉTODOS"

#End Region

#Region "FUNÇÕES ESPECIAIS"

        Private Function TestaCPF(CPF As String) As String
            Dim dadosArray() As String = {"11111111111", "22222222222", "33333333333", "44444444444'
                "55555555555", "66666666666", "77777777777", "88888888888", "99999999999"}
            Dim vResultado As Boolean = True
            Dim i, x, n1, n2 As Integer
            CPF = CPF.Trim
            For i = 0 To dadosArray.Length - 1
                If CPF.Length <> 11 Or dadosArray(i).Equals(CPF) Then
                    vResultado = False
                End If
            Next
            If vResultado = False Then
                Return "00000000000"
            Else
                For x = 0 To 1
                    n1 = 0
                    For i = 0 To 8 + x
                        n1 = n1 + Val(CPF.Substring(i, 1)) * (10 + x - i)
                    Next
                    n2 = 11 - (n1 - (Int(n1 / 11) * 11))
```

```
                    If n2 = 10 Or n2 = 11 Then n2 = 0
                    If n2 <> Val(CPF.Substring(9 + x, 1)) Then
                        vResultado = False
                        Exit For
                    End If
                Next
                If vResultado = False Then
                    Return "00000000000"
                End If
            End If
            Return CPF
        End Function

#End Region
    End Class
End Namespace
```

7) Agora, no código do formulário, dentro da sub-rotina `New()` , na mensagem de boas-vindas, escreva o CPF do cliente. Primeiramente a Gabriela:

```
Lbl_BemVindo_Gabriela.Text = "Bem Vindo " + ContaDaGabriela.Titular.Nome +
    " Agência: " + ContaDaGabriela.Agencia.ToString +
    " Conta Corrente: " + ContaDaGabriela.Conta.ToString +
    " CPF: " + ContaDaGabriela.Titular.CPF
```

8) E depois o Bruno:

```
Lbl_BemVindo_Bruno.Text = "Bem Vindo " + ContaDoBruno.Titular.Nome +
    " Agência: " + ContaDoBruno.Agencia.ToString +
    " Conta Corrente: " + ContaDoBruno.Conta.ToString +
    " CPF: " + ContaDaBruno.Titular.CPF
```

9) Para testar, no código, no momento da inicialização do CPF, digite um inválido e teste aplicação. Depois faça o mesmo usando um CPF válido.